



[> home](#) [> about](#) [> feedback](#) [> log](#)

US Patent & Trademark Office



Try the *new* Portal
design

Give us your opinion
after using it.

Search Results

Nothing Found

Your search for [(caller and (rpc or ((remote\$2 or external) <near/2> (procedure or function) <near/2> call*)) and (stack <paragraph> ((single or one or sole) <near/2> push*)))<AND>(meta_published_date <= 06-01-2001)] did not return any results.

You may revise it and try your search again below or click advanced search for more options.

```
(caller and (rpc or ((remote$2 or
external) <near/2> (procedure or
function) <near/2> call*)) and
(stack <paragraph> ((single or one
or sole) <near/2>
push*)))<AND>(meta_published_date
<= 06-01-2001 )
```



SEARCH

[[Advanced](#)

[Search](#)] [[Search Help/Tips](#)]



[Complete Search Help and Tips](#)

The following characters have specialized meaning:

Special Characters	Description
, () [These characters end a text token.
= > < !	These characters end a text token because they signify the start of a field operator. (! is special: != ends a token.)
` @ \Q < { [!	These characters signify the start of a delimited token. These are terminated by the end character associated with the start character.



[> home](#) : [> about](#) : [> feedback](#) : [> log](#)

US Patent & Trademark Office



Try the *new* Portal
design

Give us your opinion
after using it.

Search Results

Search Results for: [(((call* and stack)) <AND>((pescner:ganee or krishnaswamy or dussud)<IN> author)<AND>(meta_published_date <= 06-01-2001))]

Found 1 of 121,059 searched.

Search within Results

[> Advanced Search](#) : [> Search Help/Tips](#)

Sort by: [Title](#) [Publication](#) [Publication Date](#) [Score](#) [Binder](#)

Results 1 - 1 of 1 [short listing](#)

1 [TICLOS: an implementation of CLOS for the explorer family](#)

77%

 P. H. Dussud

**ACM SIGPLAN Notices , Conference proceedings on
Object-oriented programming systems, languages and
applications** September 1989

Volume 24 Issue 10

Results 1 - 1 of 1 [short listing](#)

The ACM Portal is published by the Association for Computing Machinery.
Copyright © 2003 ACM, Inc.



[> home](#) [> about](#) [> feedback](#) [> log](#)

US Patent & Trademark Office



Try the *new* Portal
design

Give us your opinion
after using it.

Search Results

Search Results for: [(stack <sentence> (marker or pointer or hoist*))<AND>(((caller and (rpc or ((remote\$2 or external) <near/2> (procedure or function) <near/2> call*)) and (inline or in?line or inlined or in?lined or "in line" or ((direct or directly) <near/1> call*)))<AND>(meta_published_date <= 06-01-2001)))]
Found 18 of 121,059 searched.

Search within Results



[> Advanced Search](#) [> Search Help/Tips](#)

5

Sort by: [Title](#) [Publication](#) [Publication Date](#) [Score](#) [Binder](#)

Results 1 - 18 of 18 [short listing](#)

- 1 [Efficient software-based fault isolation](#) 89%
 Robert Wahbe , Steven Lucco , Thomas E. Anderson , Susan L. Graham
ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles
December 1993
Volume 27 Issue 5
- 2 [Lightweight remote procedure call](#) 85%
 Brian N. Bershad , Thomas E. Anderson , Edward D. Lazowska , Henry M. Levy
ACM Transactions on Computer Systems (TOCS) February 1990
Volume 8 Issue 1

Lightweight Remote Procedure Call (LRPC) is a communication facility designed and optimized for communication between protection domains on the same machine. In contemporary small-kernel operating systems, existing RPC systems incur an unnecessarily high cost when used for the type of communication that predominates between protection domains on the same machine. This cost leads system designers to coalesce weakly related subsystems into the same protection domain, trading safety for ...

3 Lightweight remote procedure call

85%

 B. Bershad , T. Anderson , E. Lazowska , H. Levy

ACM SIGOPS Operating Systems Review , Proceedings of the twelfth ACM symposium on Operating systems principles


November 1989

Volume 23 Issue 5

Lightweight Remote Procedure Call (LRPC) is a communication facility designed and optimized for communication between protection domains on the same machine. In contemporary small-kernel operating systems, existing RPC systems incur an unnecessarily high cost when used for the type of communication that predominates between protection domains on the same machine. This cost leads system designers to coalesce weakly-related subsystems into the same protection domain, tradin ...

4 Fine-grain multithreading with minimal compiler support—a cost effective approach to implementing efficient multithreading languages

85%

 Kenjiro Taura , Akinori Yonezawa

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation May 1997

Volume 32 Issue 5

It is difficult to map the execution model of multithreading languages (languages which support fine-grain dynamic thread creation) onto the single stack execution model of C. Consequently, previous work on efficient multithreading uses elaborate frame formats and allocation strategy, with compilers customized for them. This paper presents an

alternative cost-effective implementation strategy for multithreading languages which can maximally exploit current sequential C compilers. We identify a s ...

5 Transaction processing monitors

84%

 Philip A. Bernstein

Communications of the ACM November 1990

Volume 33 Issue 11

A transaction processing (TP) application is a program that performs an administrative function by accessing a shared database on behalf of an on-line user. A TP system is an integrated set of products that supports TP applications. These products include both hardware, such as processors, memories, disks and communications controllers, and software such as operating systems (Oss), database management systems (DBMSs), computer networks and TP monitors. Much of the integration of these prod ...

6 Integrating segmentation and paging protection for safe, efficient and transparent software extensions

82%

 Tzi-cker Chiueh , Ganesh Venkitachalam , Prashant Pradhan

ACM SIGOPS Operating Systems Review , Proceedings of the seventeenth ACM symposium on Operating systems principles


December 1999

Volume 33 Issue 5

The trend towards extensible software architectures and component-based software development demands safe, efficient, and easy-to-use extension mechanisms to enforce protection boundaries among software modules residing in the same address space. This paper describes the design, implementation, and evaluation of a novel intra-address space protection mechanism called *Palladium*, which exploits the segmentation and paging hardware in the Intel X86 architecture and efficiently supports safe ...

7 Dynamic software updating

80%

 Michael Hicks , Jonathan T. Moore , Scott Nettles

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and

implementation May 2001
Volume 36 Issue 5

Many important applications must run continuously and without interruption, yet must be changed to fix bugs or upgrade functionality. No prior general-purpose methodology for dynamic updating achieves a practical balance between flexibility, robustness, low overhead, and ease of use.

We present a new approach for C-like languages that provides type-safe dynamic updating of native code in an extremely flexible manner (code, data, and types may be updated, at programmer-determined times ...

8 On micro-kernel construction 80%


 J. Liedtke

ACM SIGOPS Operating Systems Review , Proceedings of the fifteenth ACM symposium on Operating systems principles

December 1995

Volume 29 Issue 5


9 A language-based approach to protocol implementation 80%

 Mark B. Abbott , Larry L. Peterson

IEEE/ACM Transactions on Networking (TON) February 1993

Volume 1 Issue 1

10 Experiences with the Amoeba distributed operating system 77%

 Andrew S. Tanenbaum , Robbert van Renesse , Hans van Staveren , Gregory J. Sharp , Sape J. Mullender

Communications of the ACM December 1990


Volume 33 Issue 12

The Amoeba project is a research effort aimed at understanding how to connect multiple computers in a seamless way [16, 17, 26, 27, 31]. The basic idea is to provide the users with the illusion of a single powerful timesharing system, when, in fact, the system is implemented on a collection of machines, potentially distributed among several countries. This research has led to the design and

implementation of the Amoeba distributed operating system, which is being used as a prototype and veh ...

11 Fast procedure calls

77%


 Butler W. Lampson

Proceedings of the first international symposium on Architectural support for programming languages and operating systems March 1982

A mechanism for control transfers should handle a variety of applications (e.g., procedure calls and returns, coroutine transfers, exceptions, process switches) in a uniform way. It should also allow an implementation in which the common cases of procedure call and return are extremely fast, preferably as fast as unconditional jumps in the normal case. This paper describes such a mechanism and methods for its efficient implementation.

12 The Amber system: parallel programming on a network of

77%

 multiprocessors

J. Chase , F. Amador , E. Lazowska , H. Levy , R. Littlefield

ACM SIGOPS Operating Systems Review , Proceedings of the twelfth ACM symposium on Operating systems principles


November 1989

Volume 23 Issue 5

This paper describes a programming system called Amber that permits a single application program to use a homogeneous network of computers in a uniform way, making the network appear to the application as an integrated multiprocessor. Amber is specifically designed for high performance in the case where each node in the network is a shared-memory multiprocessor. Amber shows that support for loosely-coupled multiprocessing can be efficiently realized using an obje ...

13 Implementation of Argus

77%

 B. Liskov , D. Curtis , P. Johnson , R. Scheifer

ACM SIGOPS Operating Systems Review , Proceedings of the eleventh ACM Symposium on Operating systems principles

November 1987

Volume 21 Issue 5

Argus is a programming language and system developed to support the construction and execution of distributed programs. This paper describes the implementation of Argus, with particular emphasis on the way we implement atomic actions, because this is where Argus differs most from other implemented systems. The paper also discusses the performance of Argus. The cost of actions is quite reasonable, indicating that action systems like Argus are practical.

- 14 MIMO — An infrastructure for monitoring and managing distributed middleware environments 77%

Günther Rackl , Markus Lindermeier , Michael Rudorfer , Bernn Süss
IFIP/ACM International Conference on Distributed systems platforms April 2000

This paper presents the MIMO Middleware MOnitoring system, an infrastructure for monitoring and managing distributed, heterogeneous middleware environments. MIMO is based on a new multi-layer-monitoring approach for middleware systems, which classifies collected information using several abstraction levels. The key features of MIMO are its openness, flexibility, and extensibility. MIMO's research contribution is to enable easy integration of heterogeneous middleware platforms, to be suited ...

- 15 The performance of ?-kernel-based systems 77%


Hermann Härtig , Michael Hohmuth , Jochen Liedtke , Sebastian Schönberg
ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles
October 1997
Volume 31 Issue 5

- 16 Supporting dynamic data structures on distributed-memory machines 77%

Anne Rogers , Martin C. Carlisle , John H. Reppy , Laurie J. Hendren
ACM Transactions on Programming Languages and Systems (TOPLAS) March 1995
Volume 17 Issue 2

Compiling for distributed-memory machines has been a very active research area in recent years. Much of this work has concentrated on programs that use arrays as their primary data structures. To date, little work has been done to address the problem of supporting programs that use pointer-based dynamic data structures. The techniques developed for supporting SPMD execution of array-based programs rely on the fact that arrays are statically defined and directly addressable. Recursive data s ...

17 Sharing and protection in a single-address-space operating system 77%


 Jeffrey S. Chase , Henry M. Levy , Michael J. Feeley , Edward D. Lazowska

ACM Transactions on Computer Systems (TOCS) November 1994

Volume 12 Issue 4

This article explores memory sharing and protection support in Opal, a single-address-space operating system designed for wide-address (64-bit) architectures. Opal threads execute within protection domains in a single shared virtual address space. Sharing is simplified, because addresses are context independent. There is no loss of protection, because addressability and access are independent; the right to access a segment is determined by the protection domain in which a thread executes. T ...

18 Using continuations to implement thread management and 77%

 communication in operating systems

Richard P. Draves , Brian N. Bershad , Richard F. Rashid , Randall W. Dean

ACM SIGOPS Operating Systems Review , Proceedings of the thirteenth ACM symposium on Operating systems principles

September 1991

Volume 25 Issue 5

Results 1 - 18 of 18 short listing

The ACM Portal is published by the Association for Computing Machinery.

Copyright © 2003 ACM, Inc.

THIS PAGE BLANK (USPTO)



[> home](#) | [> about](#) | [> feedback](#) | [> log](#)

US Patent & Trademark Office



Try the *new* Portal
design

Give us your opinion
after using it.

Search Results

Nothing Found

Your search for **[((inline or inlined or in?line or in?lined or "in line" or "in lined") <near/1> stub)<AND>(((caller and (rpc or ((remote\$2 or external) <near/2> (procedure or function) <near/2> call*)) and (inline or in?line or inlined or in?lined or "in line" or ((direct or directly) <near/1> call*)))<AND>(meta_published_date <= 06-01-2001)))]** did not return any results.

You may revise it and try your search again below or click advanced search for more options.

```
((inline or inlined or in?line or
in?lined or "in line" or "in
lined") <near/1>
stub)<AND>(((caller and (rpc or
(remote$2 or external) <near/2>
(procedure or function) <near/2>
call*)) and (inline or in?line or
inlined or in?lined or "in line" or
((direct or directly) <near/1>
call*)))<AND>(meta_published_date
<= 06-01-2001 )) )
```



[\[Advanced](#)

[Search\]](#) [\[Search Help/Tips\]](#)



[Complete Search Help and Tips](#)

The following characters have specialized meaning:

Special Characters	Description
, () [These characters end a text token.
= > < !	These characters end a text token because they signify the start of a field operator. (! is special: != ends a token.)
` @ \Q < { [!	These characters signify the start of a delimited token. These are terminated by the end character associated with the start character.